

MINISTRY OF EDUCATION AND RESEARCH



---

**TECHNICAL UNIVERSITY**  
OF CLUJ-NAPOCA, ROMANIA

---

# **DOCUMENTAȚIE TEMA 1**

## **CALCULATOR DE POLINOAME**

Lupou Krisztián Róbert

Grupa 302310

Profesor Laborator: Marcel Antal

## Cuprins

1. Cerința.....	3
2. Obiective .....	3
2.1. Obiectiv Principal. ....	3
2.2. Obiective Secundare.....	3
3. Analiza Problemei.....	3
4. Proiectare.....	3
4.1. Structuri de date.....	3
4.2. Algoritmi .....	4
4.3. Interfața Utilizator.....	4
4.4. Diagrama UML.....	5
5. Implementare.....	6
6. Rezultate.....	8
7. Concluzii.....	9
8. Bibliografie.....	10

## 1.Cerința

Se cere implementarea unui calculator de polinoame care să permită următoarele operații:

- citirea a două polinoame de la tastatură;
- efectuarea operațiilor de: adunare, scădere, înmulțire, împărțire, derivare și integrare.

## 2.Obiective

### 2. 1. Obiectiv Principal

Obiectivul acestei teme este de a proiecta, de a implementa un sistem de procesare a polinoamelor cu ajutorul coeficienților și a exponenților polinoamelor. Operațiile disponibile sunt cele de adunare, scădere, înmulțire, derivare și integrare.

Pentru a înțelege funcționalitatea acestui sistem, prima dată trebuie înțeles din punct de vedere matematic termenul de polinom. Polinoamele conțin termeni numiți monoame, alcătuite dintr-o constantă, coeficient, înmulțită cu una sau mai multe variabile care pot avea exponent întreg pozitiv, numit putere sau grad al monomului în cauză.

Polinomul are forma generală:  $P(x) = a_n * x^n + a_{n-1} * x^{n-1} + \dots + a_1 * x + a_0$ .

În urma operațiilor de adunare, scădere, înmulțire, derivare și integrare rezultatul este tot un polinom.

### 2. 2. Obiective secundare

- utilizarea use case-urilor și a scenariilor: se alege operația care se efectuează asupra polinoamelor, aplicația returnând rezultatul operației dorite;
- alegerea structurilor de date: folosirea ArrayList în loc de vector pentru a opera cu această structură;
- împărțirea pe clase: am folosit clasele Monom, MonomReal, Polinom, Operații, Controller și View;
- dezvoltarea algoritmilor: algoritmi de operare cu polinoamele;
- implementarea soluției: prezentarea claselor, metodelor și a ideilor.

## 3. Analiza Problemei

Aplicația trebuie să implementeze operațiile de aduna, scădere, înmulțire, derivare și integrare, în urma introducerii celor două polinoame. În aplicație, polinomul va o listă de monoame alcătuite din coeficienți și putere. Pentru a se putea rezolva operațiile cu polinoame se respectă regulile matematice, se adună/scad monoamele de același grad, se respectă regula derivării și integrării unui monom, iar înmulțirea reprezintă o sumă de monoame înmulțite. Listele se vor parcurge pentru a se putea implementa operațiile date.

## 4. Proiectare

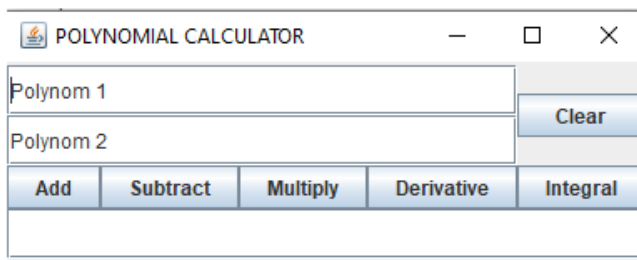
### 4. 1. Structuri de date

Am utilizat ArrayList-urile ca structură de date, în cazul polinoamelor ArrayList-urile de monoame care reprezintă polinomul. Pentru a putea folosi ArrayList-urile și funcțiile care se folosesc de liste s-a inclus biblioteca java.util.ArrayList. Din această bibliotecă am utilizat doar funcția de add.

## 4. 2. Algoritmi

Algoritmii utilizați sunt cei de calcul a polinoamelor, cum ar fi cei de adunare a polinoamelor, în care se adună polinoamele de același grad și se adaugă la rezultat cele care nu sunt comune, la fel se procedează și la scădere. La înmulțire se înmulțește monom cu monom, după care se adună monoamele obținute care au același grad. La derivare și integrare se aplică formulele corespunzătoare și se obține un alt polinom, cu observația că în cazul integrării coeficienții obținuți vor fi reali.

## 4. 3. Interfața utilizator



Interfața utilizator este reprezentată de 3 Text Field-uri dintre care două sunt utilizate pentru citirea polinoamelor sub formă de String și a treia pentru afișarea rezultatului. Pentru a utiliza Text Field-uri se importă biblioteca `javax.swing.JTextField`. Polinoamele care se introduc trebuie să fie de forma:

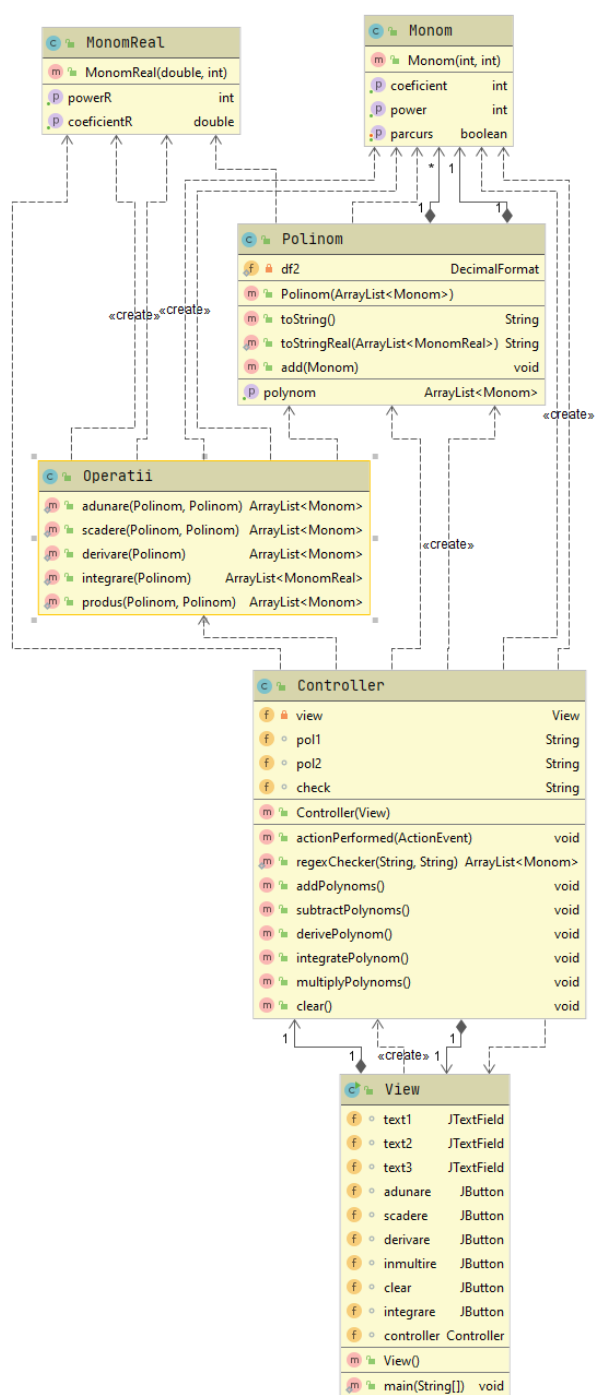
$$\pm \text{coef}(n)x^n \pm \text{coef}(n-1)x^{(n-1)} + \dots \pm \text{coef}(1)x^1 \pm \text{coef}(0)x^0$$

Polinoamele introduse pot să nu aibă unu sau mai multe monoame de grade diferite, coeficienți sunt numere întregi de la 0-9 și înaintea lor se pune semnul. Interfața dispune și de 6 butoane dintre care 5 reprezintă operațiile disponibile și un buton de clear care golește Text Field-urile. Butoanele sunt implementate cu ajutorul bibliotecilor `javax.swing.JButton`, `java.awt.event.ActionEvent`, `java.awt.event.ActionListener`. Butoanele funcționează la apăsarea click-ului stânga a mouse-ului.

#### 4. 4. Diagrama UML

Unified Modeling Language (prescurtat UML) este un limbaj standard pentru descrierea de modele și specificații pentru software. Limbajul a fost creat de către consorțiul Object Management Group (OMG) care a mai produs printre altele și standardul de schimb de mesaje între sisteme CORBA. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al cărui fundament este structurarea programelor pe clase, și instanțele acestora (numite și obiecte). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Așa se face că există aplicații ale UML-ului pentru management de proiecte, pentru *business Process Design* etc.

#### Diagrama UML a aplicației



## 5. Implementare

Clasele: Controller, View, Monom, MonomReal, Polinom și Operatii.

Clasele au fost incluse în același pachet.

Clasa Monom are atributele coeficient și power de tip întreg și parcurs de tip boolean, coeficient și power fiind coeficientul și puterea monomului. Metodele implementate în clasa Monom sunt getCoefficient(), getPower(), setParcurs() și getParcurs(), ultimele două fiind folosite pentru a indica dacă s-a găsit sau nu un monom de același grad, ele fiind folosite în cazul operațiilor de înmulțire, adunare și scădere. Constructorul clasei folosește drept parametrii coeficientul și puterea, Monom ( int coeficient, int putere). Metodele de getPower() și getCoefficient() sunt foarte utile și ajutătoare pentru manipularea caracteristicilor care definesc monomul, și anume puterea și coeficientul.

Clasa MonomReal seamănă cu clasa Monom cu diferența că ea folosește coeficient de tip double. Metodele implementate în clasa MonomReal sunt getCoefficientR() și getPowerR(), ele fiind utile în manipularea caracteristicilor care definesc un monom cu coeficient de tip real. Constructorul clasei folosește drept parametrii coeficientul și puterea, MonomReal ( double coeficient, int putere). A fost nevoie de această clasă din cauza operației de integrare, deoarece coeficientul care rezultă împărțirii dintre el și puterea + 1, și astfel rezultă un polinom cu coeficienți reali.

Clasa Polinom conține o listă de monoame care alcătuiesc polinomul în sine. Această clasă conține un constructor, Polinom(ArrayList<Monom> polynom), care primește ca parametru un polinom.

Clasa Polinom conține metodele: toString(), toStringReal(), getPolynom() și add().

Metoda getPolynom() și add() este folosit în manipularea polinoamelor, metoda add() fiind folosită pentru adăugarea a noi monoame.

Metoda toString transformă polinomul rezultat unei anumite operații într-un String care se afișează în Text Field-ul alocat rezultatului. Pentru început se verifică dacă mărimea polinomului este 0, în acest caz nu există nici un polinom, iar valoarea String-ului va fi 0. În caz contrar se intră pe cealaltă ramură. Pe aceasta se parcurge polinomul și se verifică dacă coeficientul monomului curent este mai mare decât 0, caz în care se pune “ + ” în String. Se verifică dacă coeficientul este 0, caz în care nu se scrie nimic, în caz contrar se scrie coeficientului, semnul necesar lui, a variabilei x, a căciuliței reprezentativă pentru putere ( “ ^ ” ) și a puterii.

Metoda toStringReal() este asemănătoare lui toString() cu deosebirea că este o metodă statică care are ca parametru un ArrayList de MonomReal și că este folosit în afișarea polinoamelor cu monoame cu coeficienți reali. Pentru o afișare “elegantă” am folosit o variabilă statică df2 de tipul DecimalFormat pentru a afișa coeficientul de tip double cu 2 zecimale după virgulă. Pentru DecimalFormat a fost importată biblioteca java.text.DecimalFormat.

Clasa Operații conține doar metode statice care implementează operațiile de adunare, scădere, derivare, integrare și înmulțire, fiecare fiind denumită corespunzător operației pe care este pusă să o facă.

Metoda de adunare primește ca și parametrii polinoamele citite de la tastatură, polinoame care vor fi adunate, și returnează un ArrayList<Monom>. Se parcurg cele două polinoame și unde avem monoame de același grad se adună coeficienții și se adaugă în ArrayList-ul de Monoame care se returnează monomul cu rezultatul sumei și puterea și se marchează cele două monoame ca fiind parcurse, nume\_monom.setParcurs( true ). După se parcurge separat cele două polinoame și se adaugă monoamele care nu au fost parcurse în ArrayList-ul, ele fiind monoamele care nu sunt comune.

Metoda scăderii se face la fel ca și metoda adunării, cu diferența că la monoamele de același grad se scad coeficienții care urmează să fie adăugați în monomul care se adaugă în ArrayList-ul care se returnează.

Metoda derivării primește ca parametru primul polinom care se introduce de la tastatură și returnează un `ArrayList<Monom>`. Se parcurge polinomul și monomul care se adaugă în lista care se va returna va avea coeficientul egal cu coeficientul monomului curent al polinomului folosit ca parametru înmulțit cu puterea monomului curent, iar puterea va fi puterea monomului current -1.

Metoda integrării primește ca parametru primul polinom care se introduce de la tastatură și returnează un `ArrayList<MonomReal>`. Se parcurge polinomul și monomul care se adaugă în lista care se va returna va avea coeficientul real egal cu coeficientul monomului curent al polinomului folosit ca parametru supra puterea monomului current +1, iar puterea va fi puterea monomului current +1.

Metoda înmulțirii primește ca și parametrii polinoamele citite de la tastatură, polinoame care vor fi înmulțite, și returnează un `ArrayList<Monom>`. Pentru această operație am folosit două `ArrayList`-uri, ultimul fiind cel final, care se va returna. Se parcurg cele două polinoame și în primul `ArrayList` de monoame se adaugă monoame care vor avea coeficienții rezultatul produsului coeficienților celor două polinoame, iar puterea suma puterilor monoamelor care se înmulțesc. După acest pas s-a observat problema apariției unor monoame cu același grad, nu arată prea “elegant”, așa că am parcurs `ArrayList`-ul și pe unde s-a trecut s-a marcat parcurs și unde s-a găsit monoame de același grad, ele s-au adunat, după care s-au adăugat în al doilea `ArrayList` care se returnează.

Clasa View creează interfața grafică. În clasa View se creează o fereastră principală care conține butoanele pentru fiecare operație: “Add” pentru suma, “Subtract” pentru diferență, “Multiply” pentru produs, “Derivative” pentru derivată și “Integral” pentru integrală, plus un buton “Clear” pentru golirea `TextField`-urilor, butoane care reprezintă acțiuni, și `TextField`-uri, unul pentru introducerea primului polinom, al doilea pentru a introduce cel de-al doilea polinom, și al treilea pentru afișarea rezultatului operației selectate. Pentru fiecare `TextField` avem coordonate, dimensiuni, un text inițial (doar pentru primele două, “Polinom 1” și “Polinom 2”). Fiecărui buton îi este atribuită operația specifică. În clasa Controller avem și metoda `regexChecker`. Aceasta metodă este folosită pentru a citi `String`-ul, a-i verifica lungimea. În cazul în care lungimea `String`-ului este diferită de 0 metoda aceasta cere unor argumente ale `Monom`-ului (coeficient și putere) să ia valoarea întreagă în cazul în care polinomul este scris în forma corectă. (Un exemplu relevant pentru forma corectă este:  $+3x^2 + 2x^1 + 1x^0$ , fiecare coeficient fiind însoțit în mod obligatoriu de semnul său, chiar dacă coeficientul este “+1” și fiecare putere a lui  $x$ , inclusiv 0, trebuie specificată). De asemenea, aici am implementat metoda de introducere a polinomului de la tastatură, se introduce un `String` și apoi cu funcția REGEX am “spart” `String`-ul în monoame, după un pattern corespunzător ( “ `(\\+|\\-)+[0-9]+\\^[0-9]` ” ) cu care se recunoaște din `String` care este coeficientul și care este exponentul. În pattern-ul de mai sus este dată forma monomului, după care să fie recunoscut. Primele caractere sunt pentru semn ( `// + | -` ), paranteza [ 0 – 9 ] este pentru coeficient, acesta fiind de la 0 la 9. A treia parte este pentru semnul specific puterii ( “ ^ ” ), iar ultima parte pentru întregul putere, de asemenea și el între 0 și 9.

## 6. Rezultate

### Adunarea

POLYNOMIAL CALCULATOR				
$+3x^3-2x^2+4x^1+6x^0$				Clear
$+3x^3+2x^2+2x^1+5x^0$				
Add	Subtract	Multiply	Derivative	Integral
$+6x^3+6x^1+11x^0$				

### Scăderea

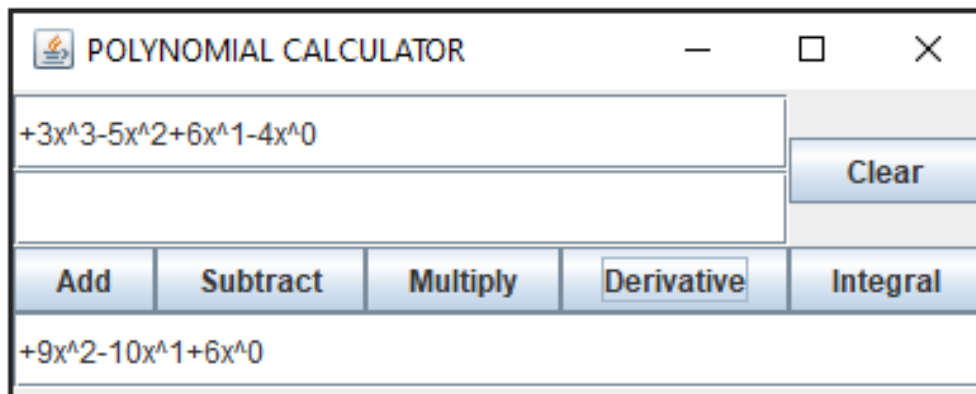
POLYNOMIAL CALCULATOR				
$+3x^3-2x^2+4x^1+6x^0$				Clear
$+3x^3+2x^2+2x^1+5x^0$				
Add	Subtract	Multiply	Derivative	Integral
$-4x^2+2x^1+1x^0$				

### Produsul

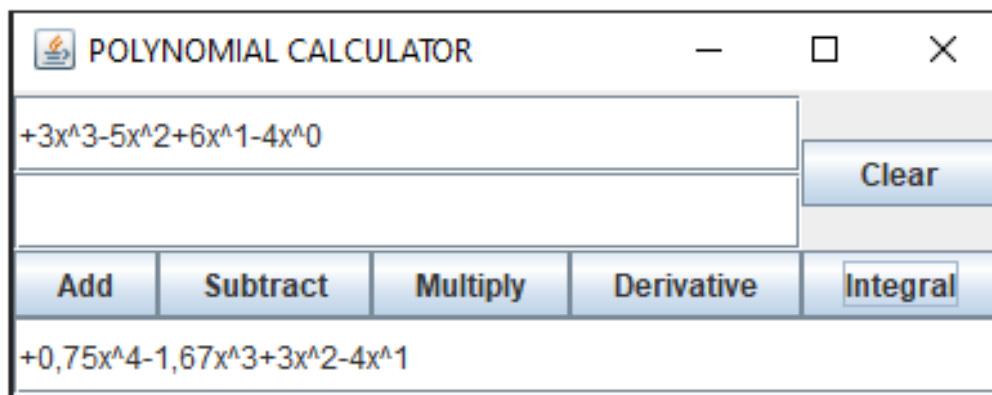
POLYNOMIAL CALCULATOR				
$+1x^2+1x^1+1x^0$				Clear
$+1x^1-1x^0$				
Add	Subtract	Multiply	Derivative	Integral
$+1x^3-1x^0$				



## Derivarea



## Integrarea



## 7. Concluzii

Îmbunătățirile care pot fi aduse acestui proiect ar putea fi împărțirea polinoamelor, aflarea rădăcinilor unui polinom, cel mai mare divizor comun a două polinoame, cel mai mic multiplu comun a două polinoame și descompunerea în factori ireductibili.

Cel mai important aspect al acestui program este funcționalitatea, modul simplu în care se pot efectua operațiile, programul având o interfață simplă potrivită utilizatorului simplu care nu are nici o idee cum se programează și pe care probabil nici nu îl va interesa, pentru el contează să funcționeze și să fie ca interfața acestui program să fie simplă, dacă vede un program cu multe comenzi de care nu are idee ce înseamnă și vede prea multe cel mai probabil se va speria de program. De aceea, este extrem de importantă eficiența programului și ușurința utilizării aplicației.

În urma acestei teme am reușit să îmi pun la punct unele aspect legate de proiectarea orientată pe obiecte și să recapitulez niște aspecte legate de polinoame. Am învățat să folosesc regex, mult mai util și mai simplu de folosit asupra String-urilor și probabil folositoare pentru proiectele viitoare.

## **8. Bibliografie**

- [https://ro.wikipedia.org/wiki/Unified\\_Modeling\\_Language](https://ro.wikipedia.org/wiki/Unified_Modeling_Language)
- <https://www.vogella.com/tutorials/JavaRegularExpressions/article.html>